

**Amendments to the Specification**

Please replace the paragraph on Page 1, lines 4 - 10 with the following marked-up replacement paragraph:

-- The present invention is related to U. S. Patent \_\_\_\_\_, titled "Machine-Oriented Extensible Document Representation and Interchange Notation" (serial number 09/\_\_\_\_); referred 09/652,056), referred to herein as the "first related invention", and U. S. Patent \_\_\_\_\_, titled "Array-Based Extensible Document Storage Format" (serial number 09/\_\_\_\_), referred 09/652,296), referred to herein as the "second related invention", both of which were filed concurrently herewith. These related inventions are commonly assigned to International Business Machines Corporation (IBM), and are hereby incorporated herein by reference. --

Please replace the paragraph that begins on Page 3, line 12 and carries over to Page 4, line 4 with the following marked-up replacement paragraph:

-- The syntax of XML is extensible and flexible, and allows document developers to create tags to convey an explicit nested tree document structure (where the structure is determined from the relationship among the tags in a particular document). Furthermore, document developers can define their own tags which may have application-specific semantics. Because of this extensibility, XML documents may be used to specify many different types of information, for use in a virtually unlimited number of contexts. It is this extensibility and flexibility which is, in large part, responsible for the popularity of XML. (A number of XML derivative notations have been defined, and continue to be defined, for particular purposes. "VoiceXML" is an example of one such derivative. References herein to "XML" are intended to

Serial No. 09/653,080

-2-

Docket RSW9-2000-0114-US1

include XML derivatives and semantically similar notations such as derivatives of the Standard Generalized Markup Language, or "SGML", from which XML was derived. Refer to ISO 8879, "Standard Generalized Markup Language (SGML)", (1986) for more information on SGML. Refer to "Extensible Markup Language (XML), W3C Recommendation 10-February-1998" which is available from the World Wide Web Consortium, or "W3C", for on the World Wide Web at <http://www.w3.org/TR/1998/REC-xml-19980210>, for more information on XML.) --

Please replace the paragraph on Page 6, lines 11 - 18 with the following marked-up replacement paragraph:

-- (1) The parser operating on the source document spends a considerable amount of effort understanding the content and meaning of the data from the XML format (as mentioned above). For example, it scans every tag thoroughly to figure out the information needed to construct a Document Object Model ("DOM") tree, upon which existing stylesheet engines operate. (DOM is published as a Recommendation of the W3C, titled World Wide Web Consortium ("W3C"), titled "Document Object Model (DOM) Level 1 Specification, Version 1.0" (1998) and available on the Web at <http://www.w3.org/TR/REC-DOM-Level-1>. "DOM" is a trademark of Massachusetts Institute of Technology.) --

Please replace the paragraph on Page 21, lines 5 - 11 with the following marked-up replacement paragraph:

-- With more and more application programs being written to operate upon XML documents, the improvements yielded by this high-performance transformation technique will

have a significant impact. (The high-performance transformation technique disclosed herein may also be used advantageously for transforming documents that have been encoded in other structured document notations, and thus references herein to using the transformation technique of the present invention for XML and mXML documents is intended documents are intended for purposes of illustration and not of limitation.) --

Please replace the paragraph that begins on Page 21, line 14 and carries over to Page 22, line 7 with the following marked-up replacement paragraph:

-- Fig. 4A illustrates a simple structured document 400 which is represented in the existing XML notation. This document contains 6 elements which are organized in a 3-level hierarchy. The node having element name "root\_element" 402 is the root node, being at the highest level of the hierarchy. This node has 2 child nodes, having element names "level\_one\_element1" 410 and "level\_one\_element2" 420. Node "level\_one\_element1" 410 also has 2 child nodes, which are the nodes having element names "level\_two\_element11" 412 and "level\_two\_element12" 414, and ~~node "level\_two\_element22" 420~~ node "level\_one\_element2" 420 has a single child node having element name "level\_two\_element21" 422. A tree structure 430 representing document 400 is shown in Fig. 4B, where the tags for the 6 elements are depicted inside rectangular shapes representing nodes of the tree and the data content corresponding to each node is shown inside an ellipse. This interpretation of an XML document 400 and its corresponding tree structure 430 are well known in the art. --

Please replace the paragraph on Page 27, lines 6 - 18 with the following marked-up replacement

Serial No. 09/653,080

-4-

Docket RSW9-2000-0114-US1

paragraph:

-- Fig. 7 provides an example of a map 700 that may be used to rename a node in a document. Suppose that the source document is the set of names shown in Fig. 1. For a particular business application, it may be desirable to replace the element name "LAST\_NAME" (see 105, 110, [[135,]] 130, 140) with the element name "SURNAME" when creating an output document. Thus, this example map 700 specifies to search for nodes having this LAST\_NAME element name and -- using the short transformations of the present invention -- renames those nodes when creating the output document. As shown in this example, the attribute named "opcode" 705 has a value of "RENAME" 710 and a parameter 715 which includes a source node 720 and a target node 740. The element name to be replaced is identified within the source tag pair 720, 735 by providing, as the value of a "tag" attribute 725 on an XMLDATA tag, the name 730 of the element in the source document. A target node tag pair 740, 750 is provided to specify a target for the map operation. A child node having the element name "tag" identifies the element name 745 to use in the output document. --

Please replace the paragraph that begins on Page 31, line 10 and carries over to Page 32, line 7 with the following marked-up replacement paragraph:

-- Fig. 11 depicts an overview of the processing that occurs when an arbitrary incoming document arrives at run time, and is to be transformed. At Block 1100, an incoming document, which in the preferred embodiment is an XML document, arrives. In Block 1110, this [[This]] document is programmatically scanned and compared to the signatures which have been previously created using the logic of Fig. 10, thereby determining whether this document contains

any of the predetermined well-defined elements that are suitable for short transformations.

Because a signature typically represents only a subset of the tags in a source document type, it is possible that no matching signature is found (in which case the processing of Fig. 11 ends, and an error condition is preferably returned to the caller), or that more than one matching signature is found. In the latter case, a conflict resolution technique which is suitable to the needs of a particular environment may be used to select from among the transformation objects. When both short and long transformations apply, the short transformations preferably take precedence over the long transformations. If multiple short transformations apply, then a technique for selecting among those is used. For example, the signature having the most matching elements (which is therefore a more precise match) may be preferred. Or, elements may be weighted, such that the signature with the highest weight is selected. As another alternative, explicit priority values may be set by the creators of signatures, in which case the matching signature with the highest priority is selected. Or, dynamic conditions such as the time of day or processing load might influence the selection. Many other conflict resolution techniques can be imagined. --

Please replace the paragraph that begins on Page 35, line 19 and carries over to Page 36, line 8 with the following marked-up replacement paragraph:

-- Fig. 14A depicts a preferred embodiment of a general representation of the logic which may be used to perform any one of the core transformations described above by directly accessing the arrays created according to the second related invention. A typical process begins by locating the element having a particular name by searching through the element name array (Block 1400 Block 1400), and then obtaining (Block 1405) the node index of the element array entry with the

matching name. Block 1410 gathers the information about this element, according to the type of short transformation being performed. This information is retrieved easily and efficiently using the located node index to index into the other arrays. An API (Application Programming Interface) is preferably invoked, as shown at Block 1415, to perform the particular transformation on this element's information. --

Please replace the paragraph that begins on Page 36, line 9 and carries over to Page 37, line 4 with the following marked-up replacement paragraph:

-- Suppose the desired high-speed transformation specified by a map is deleting an element from a document. The logic shown in the flowchart of Fig. 14B may be used to perform this delete element operation. The process begins by searching through the element name array to locate the element having a particular name (Block 1420). This comprises using the starting and offset values, along with the buffer pointer, to find the text of the element name, and comparing that text to the particular name of interest. (Alternatively, the node index may be known in some cases, in which this search is not required.) Once the name is located, the corresponding node index is then available. (In some cases, multiple array entries may have the matching name, and the indexes index of each such entry is then obtained, for example by building a list of indexes, in Block 1420.) The index of this element's parent is then obtained (Block 1425) by using the node index to access the parent array. The parent's index is then used (Block 1430) to access the children list array. The index of the element being deleted is then removed from the list for this parent. With reference to Fig. 5E, for example, to delete the element having the name "level\_one\_element2", which is shown at 420 in Fig. 4A, this node name is found in the array

name entry having index 4. Using index 4 to access the parent array indicates that this element's parent has the index 0 (see 553 of Fig. 5D). Next using index 0 to index the children list array, the index value 4 is deleted from the list at 570. --

Please replace the paragraph that begins on Page 39, line 11 and carries over to Page 40, line 8 with the following marked-up replacement paragraph:

-- This coexistence approach is depicted in Fig. 15, and includes a special parser that will parse the input XML data directly from the existing human readable format (or, alternatively, other formats such as mXML may be used) into the new array-based internal data structures; a set of maps (or "canned" XSLT stylesheets) defining a core set of transformations for reformatting the input XML on the fly; a special map-based transformation engine, optimized for processing the maps; a development process that enables binding (i) the map-based transformations such that they identify the need to use the special map-based processor and (ii) other transformations such that they identify the need to use the general XSLT engine; and an optional aggregation method that allows the combination of multiple map-based transformations into a type of "super transformation" (such as the example shown in Fig. 9) that may be performed in one operation thus avoiding the overhead of repetitive parsing. Fig. 15 illustrates this architecture as what may be termed "XML Transformation ~~Callable~~ Services Manager", where short transformations are handled using the techniques disclosed herein and long transformations are handled using standard XSLT processing. The maps and XSLTs are prebundled in transformation processing objects ready to execute ~~when matching~~ when a matching input ~~documents~~ document (such as an XML document) is received. Maps can also be preprocessed from canned XSLTs. The net effect of

this technique is to provide a full-function XSLT processing method for a spectrum of transformation performance objectives. Both XSLT flexibility and specialized transformations are considered. The callable services may be directly invoked from transaction programs through a set of transformation APIs. --